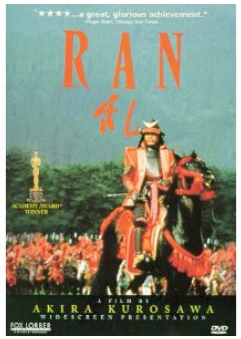
+

5 Handouts Today:

- Syllabus
- Lecture notes
- HW 1
- Handout 1: Good Writing, Induction, and Greed **(please read before doing HW 1)**
- Blank worksheet

"It is a mistake to think that you can solve any major problems just with potatoes" – Douglas Adams
"Of all the courses we've ever taken, this was one of them!" -Rotten Tomatoes
"We don't have enough words to express our feelings about this course." – Ebert & Roeper

Course staff:



- Professor: Ran “RON” Libeskind-Hadas

- Grutors: **16** friendly grutors! (names on syllabus)

- Aliens: 1

Give us a “Snappy”

On your worksheet…

- Your name
- Something that you'd like to learn about in this course
- What are your thoughts for life after college? (If it exists!)
- A random fact about you!

Learning Objectives

- Given a challenging computational problem:
  - Design an efficient algorithm for it
  - Derive its worst-case running time
  - Prove the correctness of the algorithm
- Get comfortable using a number of major design paradigms and techniques:
  - Greed, divide-and-conquer, dynamic programming
  - Amortized analysis
  - Graph algorithms
  - NP-completeness and approximation algorithms

Learning Objectives

- See some "advanced topics" to get a sense of the breadth and depth of the field
  - Parallel algorithms (~~the future!~~ the present!)
  - Computational geometry (computer graphics games)
- Good writing!
  - Read Handout 1
  - A significant % of the points are for good writing

+ min is  tion

"Everything" is on Canvas!

- Syllabus
- Homework assignments
- Submission system
- Lecture notes
- Exam dates
- Discussions
- Q & A with Ran and Grutors **(instead of e-mail!)**
- Grades

Please read the syllabus

In a nutshell…

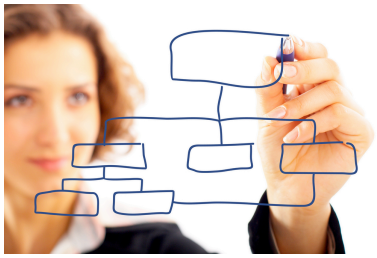
- HWs (40% **100 pts/week**)
  - Typically due on Tuesdays at 11:59 PM
  - Always submit pdfs
  - First two weeks, use LaTeX (resources in Syllabus, source code provided for each problem)
  - May submit a problem 24 hours late for a 5 point deduction
  - You get 5 free late day rubles in your "account"
- Worksheets everyday in class (10% **10 homework points each!**)
- Two "midterm" exams (15% each **takehomes in lieu of assignments**)
- In-class final (20%)
- Grading on "fixed scale" **(12 points per letter grade, e.g. 88-100 is A- to A+)**

Homework Collaboration

- Homework
  - You may talk and interact at white board with…
    - Students who are *currently* in this class
    - The grutors
    - Ran
  - You may use the book and lecture notes
  - You should **not** look for solutions…
    - On the web
    - In other books
    - From people other than those listed above
    - Anywhere else
  - Questions? Concerns? Talk to Ran!

Bonus Problems!

- You can  the course without them…
- But, they are great for…
  - Learning the material more deeply



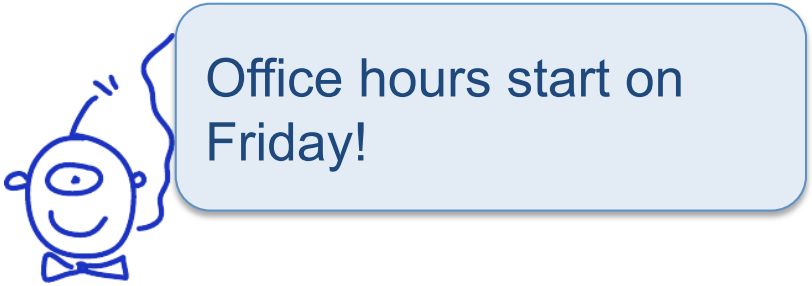
  - Being intellectually challenged

  - Putting some homework points in the

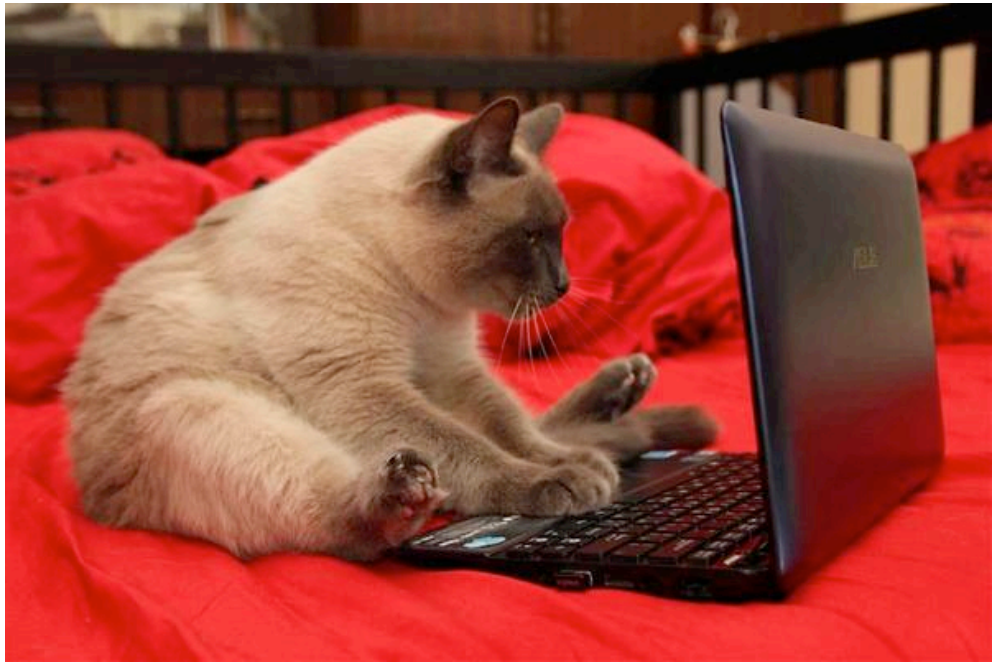
Office Hours and Grutor Hours

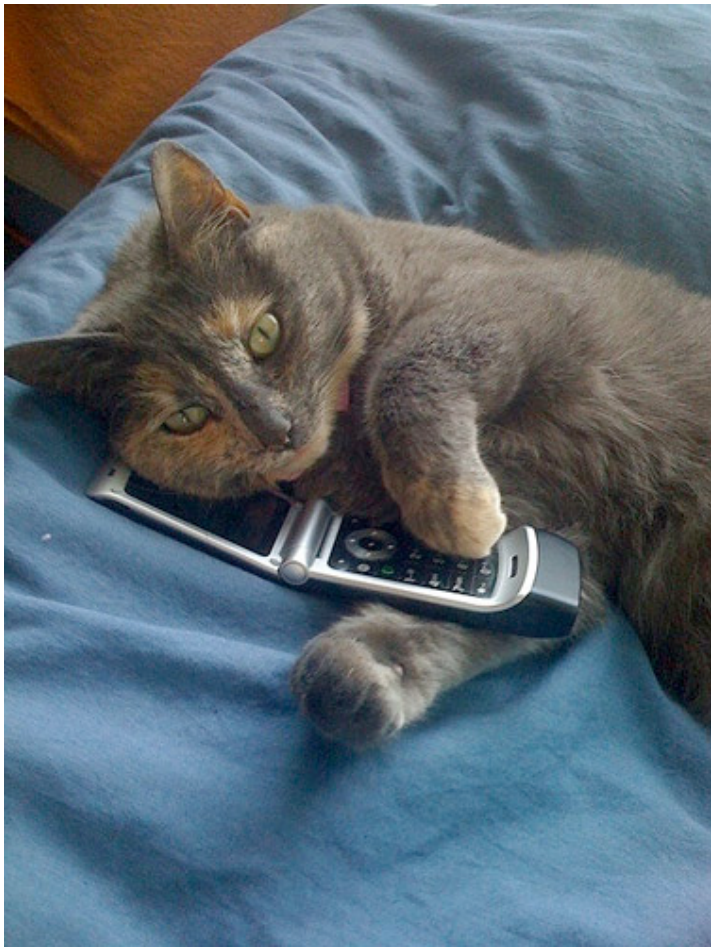
(Linked from Canvas Syllabus)

- Office hours (Olin 1253) start this Friday!
  - Mondays 4-5 PM
  - Tuesdays 11-12:30 PM (may change in February)
  - Fridays 4-5 PM
- Grutor hours (Starting this Sunday; see the link from the Syllabus on Canvas)
- Short questions/clarifications using Canvas



Pleze Not Use in Class…



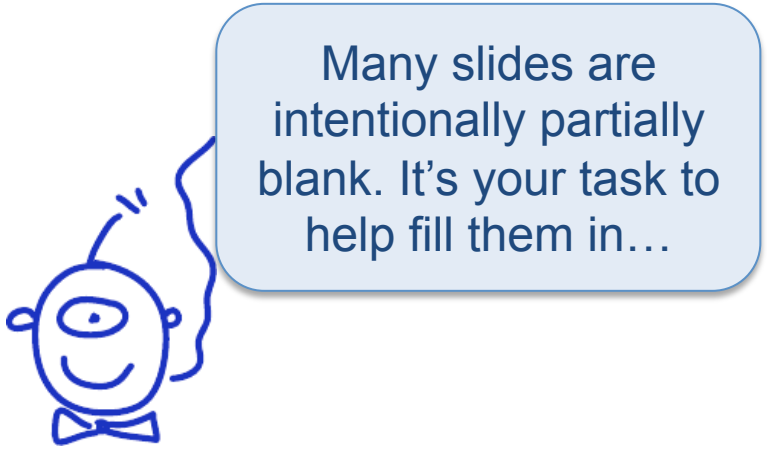


And not this…

A Note About Notes…

Theorem: One-eyed aliens are cute.

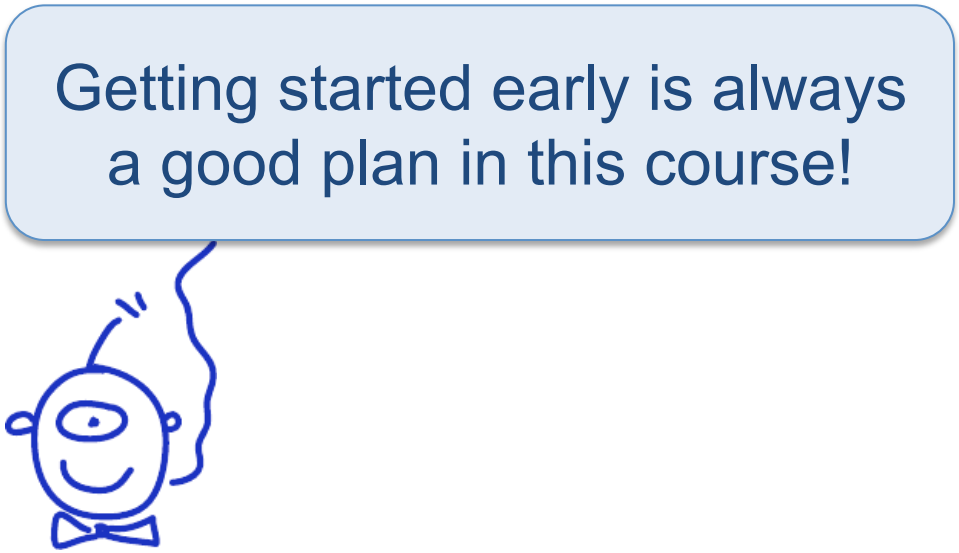
Proof: Fill this in!



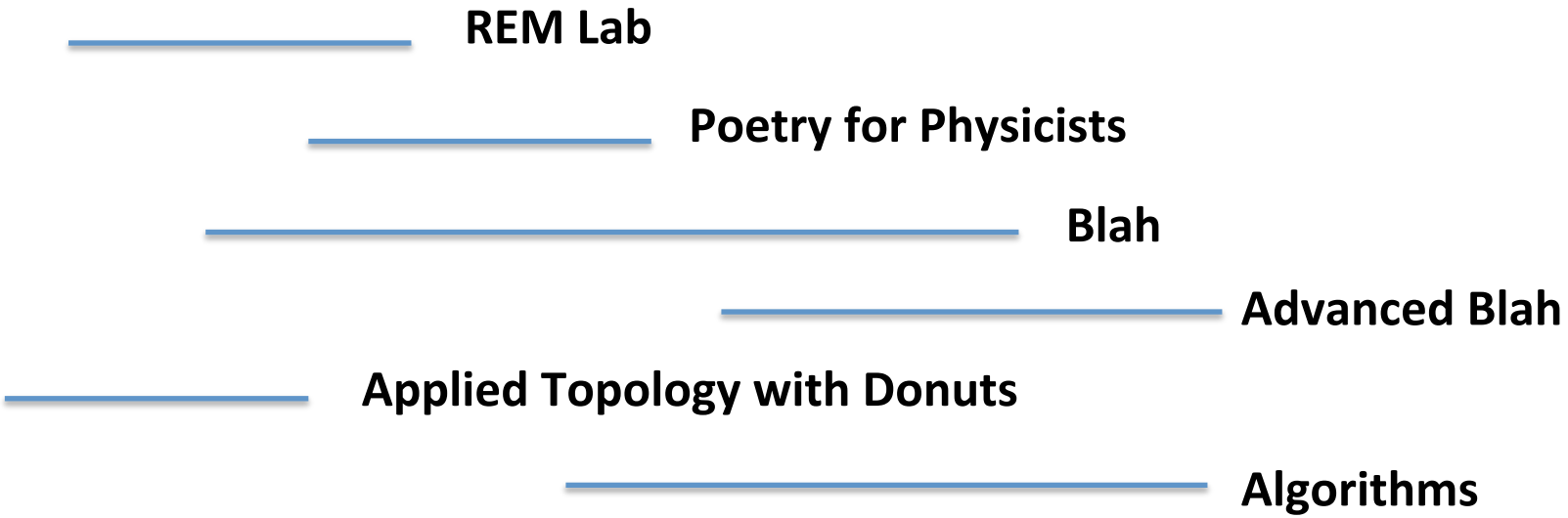
Give Me Greed, Now!*

* Use greed with extreme caution

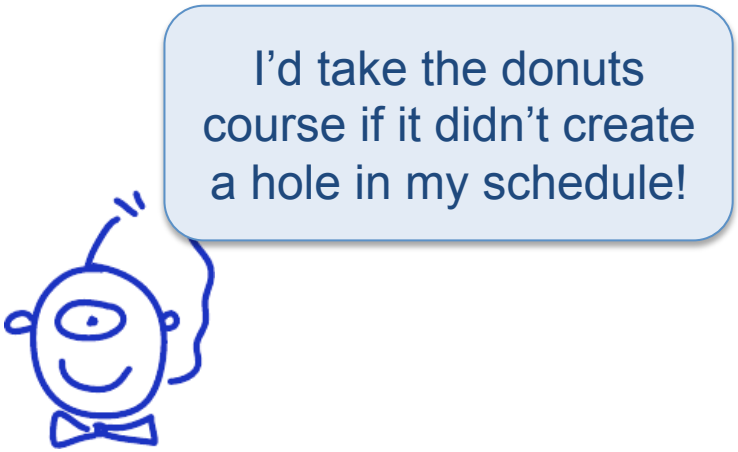
After today's class, you'll have all you need to do *all* of Homework 1!



The Pasadena Institute of Technology (P.I.T.) Registration Problem



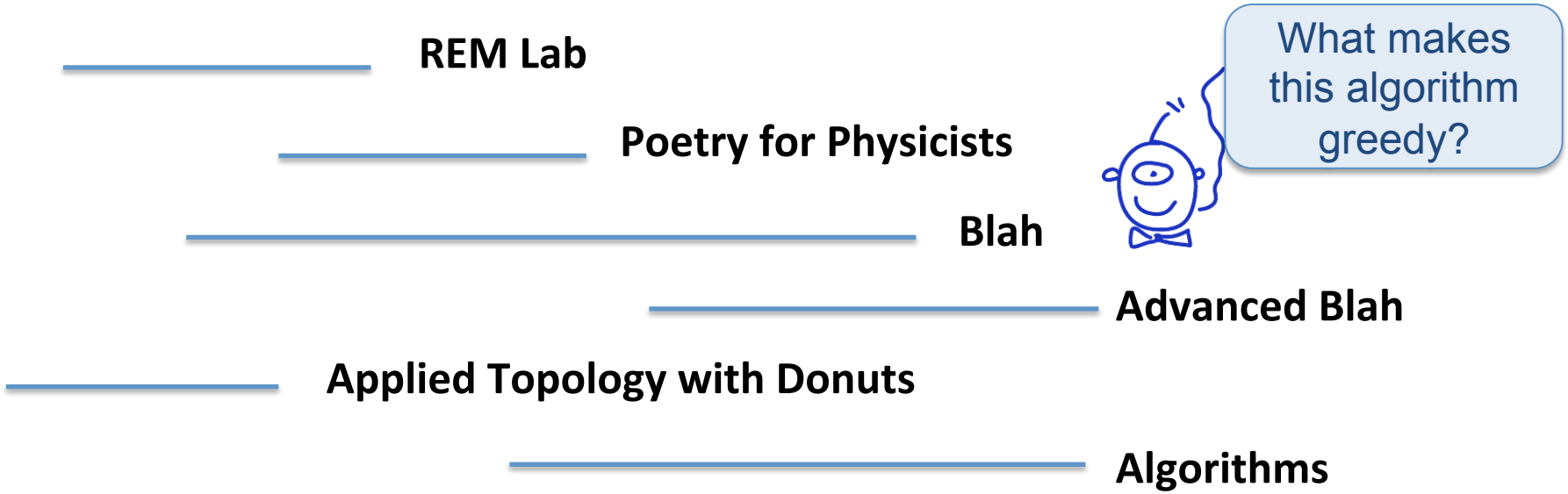
Goal: Choose a largest possible set of non-intersecting courses (there may be many optimal solutions, we just seek one!)



Prof. I. Lai's Greedy Algorithms

Attempt 1: Choose the shortest interval (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

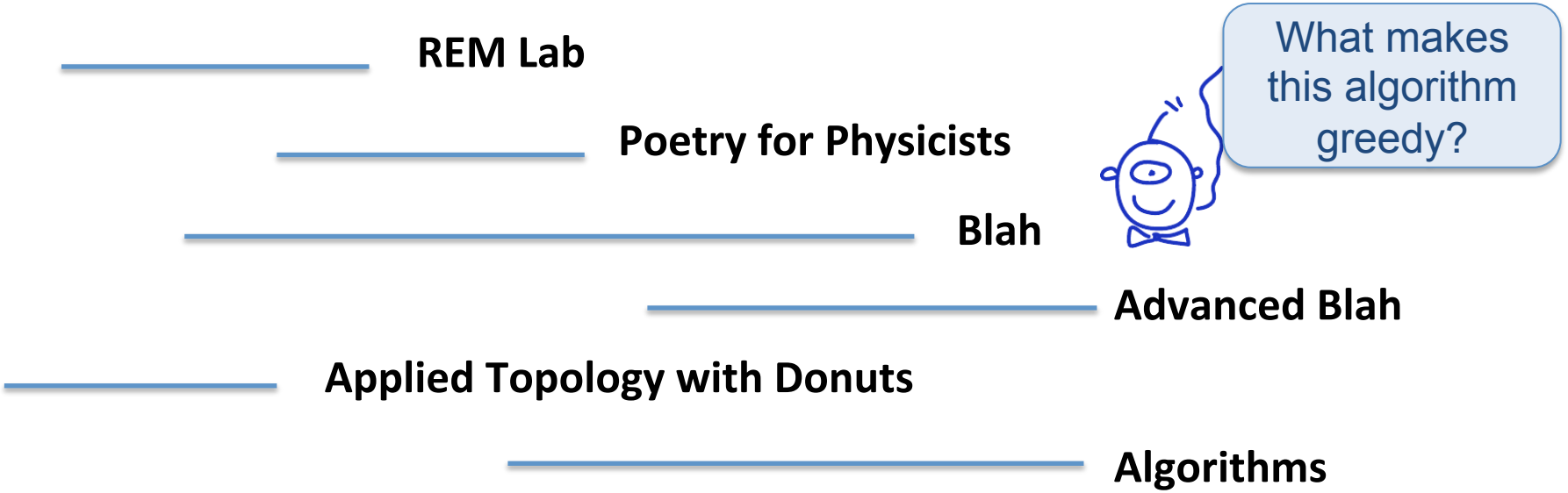
Prof. I. Lai
P.I.T



Prof. I. Lai's Greedy Algorithms

Attempt 2: Choose the interval that overlaps with the fewest other intervals (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

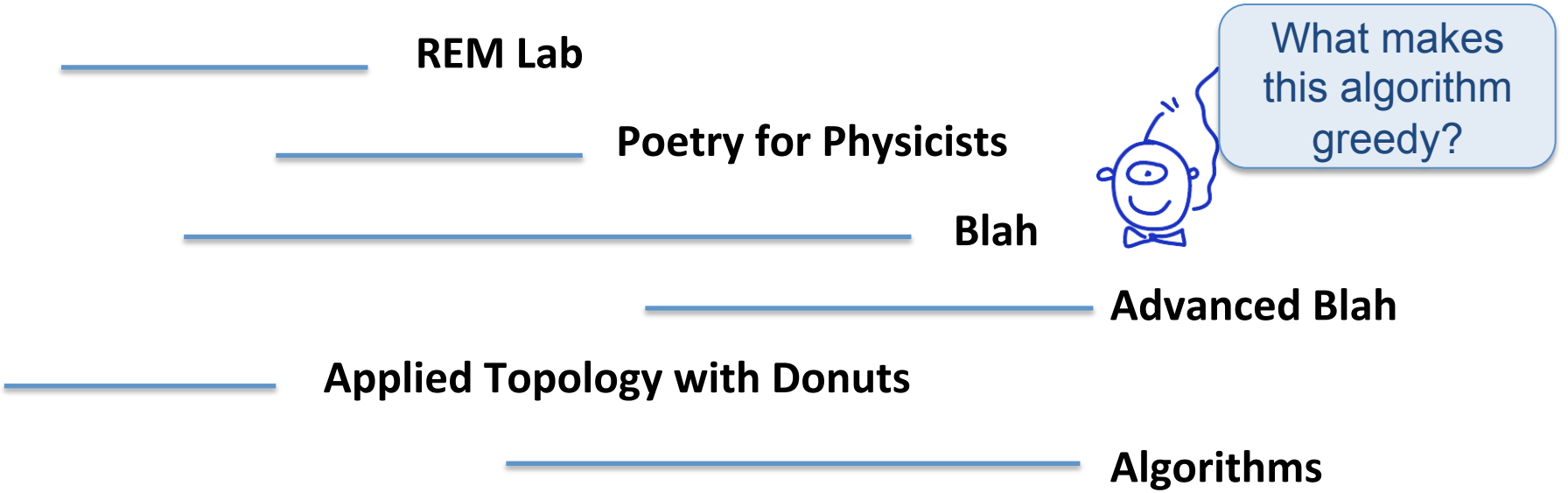
Prof. I. Lai
P.I.T



Prof. I. Lai's Greedy Algorithms

Attempt 3: Choose the interval that starts the earliest (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T



Prof. I. Lai's Greedy Algorithms

Attempt 1: Choose the shortest interval (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T

Prof. I. Lai's Greedy Algorithms

Attempt 1: Choose the shortest interval (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T

Prof. I. Lai's Greedy Algorithms

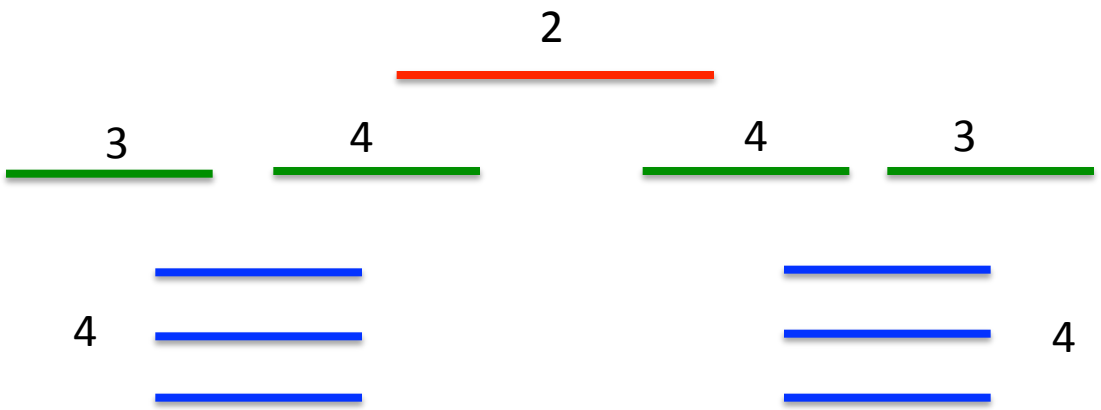
Attempt 2: Choose the interval that overlaps with the fewest other intervals (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T

Prof. I. Lai's Greedy Algorithms

Attempt 2: Choose the interval that overlaps with the fewest other intervals (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T



Prof. I. Lai's Greedy Algorithms

Attempt 3: Choose the interval that starts the earliest (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T

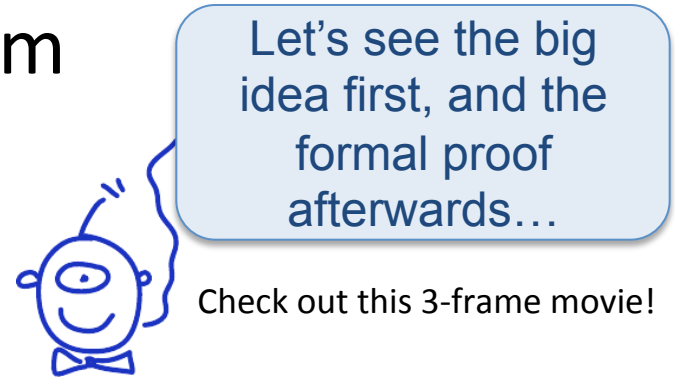
Prof. I. Lai's Greedy Algorithms

Attempt 3: Choose the interval that starts the earliest (breaking ties arbitrarily), take it, remove overlaps, recurse on remaining problem

Prof. I. Lai
P.I.T

A Correct Greedy Algorithm "Earliest Ending Time (EET)"

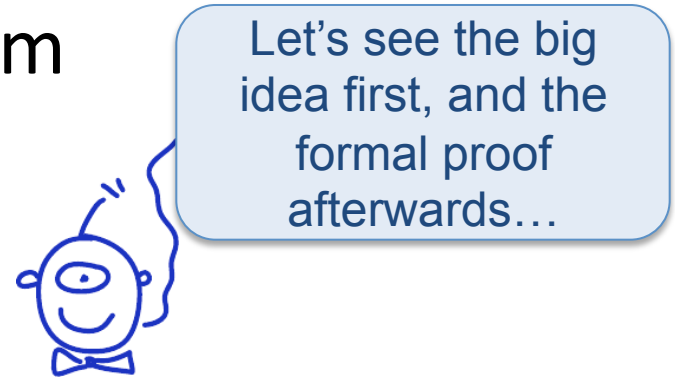
- Sort the intervals by *ending time*
- Greedily take the interval that ends first (use it!)
- Remove the intervals that overlap with the one just selected
- Recurse on remaining problem



Check out this 3-frame movie!

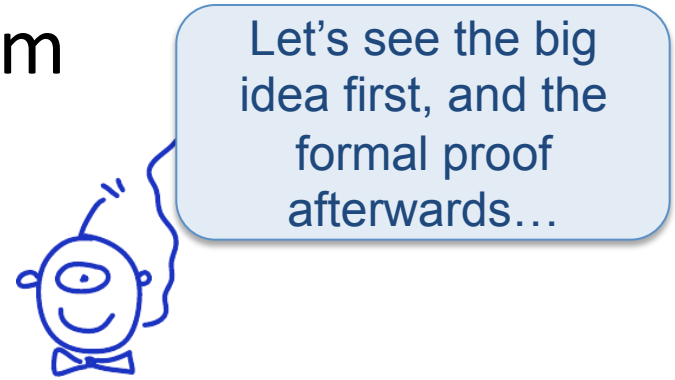
A Correct Greedy Algorithm "Earliest Ending Time (EET)"

- Sort the intervals by ending time
- Greedily take the interval that ends first (use it!)
- Remove the intervals that overlap with the one just selected
- Recurse on remaining problem



A Correct Greedy Algorithm "Earliest Ending Time (EET)"

- Sort the intervals by ending time
- Greedily take the interval that ends first (use it!)
- Remove the intervals that overlap with the one just selected
- Recurse on remaining problem



Proving that the "EET Greedy Algorithm" is Correct: Safety and Induction!

Claim: The EET greedy algorithm finds a solution with the largest number of non-overlapping intervals.

Proof: Strong induction on the number of intervals, n.

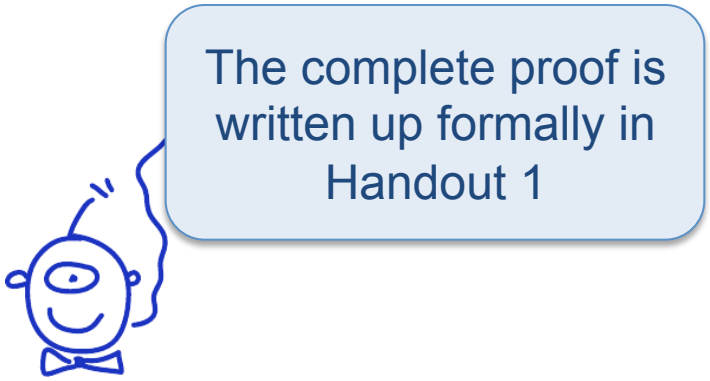
There may be multiple optimal solutions. We just seek one!

Basis: n = 0

Induction hypothesis: Assume the claim is true for *any* set with *any* number of intervals between 0 and n.

Induction Step: Consider any set of n+1 intervals. We wish to show that the EET greedy algorithm finds a solution with the largest number of non-overlapping intervals.

Finishing up the proof…



Making…

Change

```
>>> change(42, [25, 10, 5, 1])
5
>>> change(42, [10, 5, 1])
6
```

Pre-1971 British System:

1, 3, 6, 12, 24, 30, 60, 240

```
>>> change(48,  [1, 3, 6, 12, 24, 30, 60, 240])
2
```

but greedy algorithm uses…

In this problem, we're allowed to use a coin denomination as many times as we want!

Making Change!

Return the least number of coins required to make up the given amount

```
>>> change(42, [1, 5, 10, 25, 50])
```

```
change(amount, coins[1,…,i] )
  if amount == 0: return 0
  elif i == 0: return Infinity
  elif coins[i] > amount:
     return change(amount, coins[1,…,i-1])
  else:
     useIt = 1 + change(amount-coins[i],
                        coins[1,…,i])
     loseIt = change(amount, coins[1,…,i-1])
     return min(useIt, loseIt)
```

This is NOT a greedy algorithm because it's not using a simple (aka "greedy") rule to choose a coin and then move on!

Warning: Greed is generally bad!

But it works for some cases…

Consecutive powers of b

Consider denominations of type b^0, b^1, …, b^k for some positive integer $b > 1$ and $k \geq 0$.

change(amount) # greedy version for coins 1, b, b^2, …, b^k

- **Choose the largest coin b^i that doesn't exceed amount**
- **Recurse on change(amount-b^i)**

The Greedy Algorithm…

Consider denominations of type b^0, b^1, …, b^k for some positive integer $b > 1$ and $k \geq 0$.

Observation 0: Geometric series: $b^0 + \ldots + b^j = \ldots$

Observation 1: It is never optimal to use b or more coins of value b^j if $j < k$.

Observation 2: Using at most (b-1) coins of each type b^0, …, b^{i-1} we can make up at most…

Proving Correctness of the Greedy Algorithms

Proof by strong induction on amount, n

(not on our coin set 1, b, b^2, …, b^k)

Basis: n = 0

Induction hypothesis: Assume that the greedy algorithm uses the optimal number of coins for any amount from 0 to n.

Induction step: Consider an amount n+1. The greedy algorithm uses the largest coin b^i (i between 0 and k) that doesn't exceed n+1. We **first** claim that this choice is ***safe*** in the sense that there exists an optimal solution that uses a b^i coin.

Consider an optimal solution S (a multiset of coins) for amount n+1.

Observation 1: It is never optimal to use b or more coins of value b^j if $j < k$.
Observation 2: Using at most (b-1) coins of each type b^0, …, b^{i-1} we can make up at most $b^i - 1$

For what denominations does the greedy algorithm give optimal solutions?

Optimal Bounds for the Change-Making Problem

Dexter Kozen[1] and Shmuel Zaks[2]

[1] Computer Science Department, Cornell University
Ithaca, New York 14853, USA
kozen@cs.cornell.edu

[2] Computer Science Department, Technion
Haifa, Israel
zaks@cs.technion.ac.il

Abstract. The *change-making problem* is the problem of representing a given value with the fewest coins possible. We investigate the problem of determining whether the greedy algorithm produces an optimal representation of all amounts for a given set of coin denominations

Fibonacci denominations work!